

# Towards Parallelizable Sampling-based Nonlinear Model Predictive Control

R.V. Bobiti \* M. Lazar \*

\* *Department of Electrical Engineering, Eindhoven University of Technology,  
The Netherlands (e-mails: r.v.bobiti@tue.nl, m.lazar@tue.nl)*

**Abstract:** This paper proposes a new sampling-based nonlinear model predictive control (MPC) algorithm, with a bound on complexity quadratic in the prediction horizon  $N$  and linear in the number of samples. The idea of the proposed algorithm is to use the sequence of predicted inputs from the previous time step as a warm start, and to iteratively update this sequence by changing its elements one by one, starting from the last predicted input and ending with the first predicted input. This strategy, which resembles the dynamic programming principle, allows for parallelization up to a certain level and yields a suboptimal nonlinear MPC algorithm with guaranteed recursive feasibility, stability and improved cost function at every iteration, which is suitable for real-time implementation. The complexity of the algorithm per each time step in the prediction horizon depends only on the horizon, the number of samples and parallel threads, and it is independent of the measured system state. Comparisons with the *fmincon* nonlinear optimization solver on benchmark examples indicate that as the simulation time progresses, the proposed algorithm converges rapidly to the “optimal” solution, even when using a small number of samples.

**Keywords:** suboptimal nonlinear predictive control, sampling-based optimization, embedded model predictive control, dynamic programming, constrained control

## 1. INTRODUCTION

A vast literature on nonlinear model predictive control (NMPC) has proven both the theoretical (Grüne and Pannek, 2011), as well as the practical advantage (Magni et al., 2009) of this method in treating optimally the control of multi-variable nonlinear systems subject to constraints on state and inputs. Common research interests include methods for reducing the complexity of the NMPC algorithms, to make them applicable on devices of low memory (ASIC, FPGA), such as explicit NMPC (ENMPC), see, e.g., (Johansen, 2004). Much work has been done on treating other limiting factors, such as the requirement of NMPC of solving an optimization problem online. This is not well achieved by common optimization tools, which have no specific termination time, especially due to non-convexity which involves multiple local-minima. Therefore, real time requirements are not met, which limits the industrial impact of NMPC. To alleviate these concerns, the literature has proposed multiple solutions, such as approximate dynamic programming (DP) (Bertsekas, 2005a), suboptimal MPC (Scokaert et al., 1999), approximative DP and suboptimal control via rollout algorithms (Bertsekas, 2005b), NMPC based on system approximation by neural models (Ławryńczuk, 2009).

An alternative strategy in NMPC is to draw samples from either the state or input space, to design computationally feasible NMPC methods, see for example, (Piovesan and Tanner, 2009) which proposes a randomized approach to sampling the space of predicted input sequences. More recently, in (Chakrabarty et al., 2016), an ENMPC method was proposed based on sampling of the state space for continuously differentiable nonlinear systems. The method therein solves optimization problems offline to find optimal control sequences, which are used to construct the ENMPC strategy. While there are still

concerns in ENMPC related to robustness, feasibility of the offline optimization and finding the neighbors in the sampled grid for the off-grid states, ENMPC, when successful, reduces significantly the computational load of MPC at the expense of an acceptable cost degradation.

Input and state space sampling-methods for solving NMPC via approximate DP have also been proposed, see (Bertsekas, 2005a), though they inherit the dimensionality issues of DP (Lee and Lee, 2004).

Another relevant sampling-based strategy, the so-called sampling based MPC (SBMPC), was proposed in (Dunlap et al., 2010). The method therein is applicable to nonlinear systems in general, though, its performance is dependent on a user-specified heuristic. A more ample discussion on sampling-based DP and SBMPC, in the light of the method proposed in this paper is reported in Section 2.3.

A common problem of sampling-guided methods for NMPC is the sampling strategy. For example, with each input sample, a tree is expanded. After the tree is built, the path of least cost in the tree is selected from the initial state to the desired state. If the sampling is performed over the input space, and each sample is connected to all the samples in the input space for the next time step in the control horizon, then the tree growth is exponential with the horizon. Alternatively, as in randomized MPC (Piovesan and Tanner, 2009), sampling randomly in the input space, of dimension  $m$ , augmented to the horizon of dimension  $N$  requires a large number of samples, in an  $mN$  dimensional space, to achieve a significant accuracy.

In this paper we adopt a suboptimal formulation of NMPC, as originally proposed in (Scokaert et al., 1999), where it was shown that feasibility of a solution implies stability under

suitable conditions. This, together with the fact that suboptimal NMPC has the same inherent robustness properties as optimal NMPC, see (Pannocchia et al., 2011) and (Lazar and Heemels, 2009), suggest that suboptimal NMPC is a viable and in fact the best one can hope for when a sampling-guided MPC strategy is undertaken for the control of nonlinear systems. Furthermore, we aim at a sampling method which provides a suboptimal solution that yields good control performance, has a reasonable computational complexity increase with the prediction horizon and allows for parallel implementation up to some level.

In this paper, the main idea for achieving this goal is to use the shifted sequence of predicted inputs from the previous time step as a warm start, and to iteratively update this sequence by changing its elements one by one, starting from the last predicted input and ending with the first predicted input. This strategy resembles the dynamic programming principle, especially the rollout algorithms, see (Bertsekas, 2005b), which improve a heuristic base policy for optimal control. Additionally, in this paper, we sample the original input space, which is typically represented by a proper set  $\mathbb{U} \subset \mathbb{R}^m$ . Sampling allows for parallelization of the calculations performed for updating each of the elements of the predicted sequence of inputs and it enables limiting the computational time according to the requirements of the considered application. An upper-bound on the complexity of the overall algorithm is quadratic with the prediction horizon  $N$  and linear with the number of samples in  $\mathbb{U}$ . This enables the usage of long prediction horizons or real-time implementation on inexpensive computing devices such as ASIC and FPGA. The suitability for real-time implementation is also enhanced by the fact that the algorithm can be stopped at any iteration performed within a sampling period, while the complexity of the calculations per iteration depends only on the horizon  $N$ , the number of samples and parallel threads, and it does not depend on the measured state of the system. Moreover, the updated predicted sequence of inputs obtained at any iteration will guarantee recursive feasibility, stability and an improved cost function under the same conditions as in suboptimal NMPC (Scokaert et al., 1999).

The remainder of this paper is organized as follows. In Section 2, basic notation is introduced, together with the problem formulation and a discussion on the relation with the existing methods. Section 3 presents the main result as a prototype algorithm and its complexity analysis. In Section 4, three examples illustrate the potential of the developed method, and Section 5 concludes the paper.

## 2. PRELIMINARIES

### 2.1 Notation

Let  $\mathbb{R}$ ,  $\mathbb{R}_+$ ,  $\mathbb{Z}$  and  $\mathbb{Z}_+$  denote the field of real numbers, the set of non-negative reals, the set of integers and the set of non-negative integers, respectively. For every  $c \in \mathbb{R}$  and  $\Pi \subseteq \mathbb{R}$ , define  $\Pi_{\geq c} := \{k \in \Pi \mid k \geq c\}$  and similarly  $\Pi_{\leq c}$ . Let  $\text{int}(\mathbb{S})$  denote the interior of a set  $\mathbb{S}$ . Let  $\mathbb{S}^h := \mathbb{S} \times \dots \times \mathbb{S}$  for any  $h \in \mathbb{Z}_{\geq 1}$  denote the  $h$ -times Cartesian-product of  $\mathbb{S} \subseteq \mathbb{R}^n$ . A set  $\mathcal{S} \subset \mathbb{R}^n$  is called proper if it is non-empty, compact and  $0 \in \text{int}(\mathcal{S})$ .

For a vector  $x \in \mathbb{R}^n$ , the symbol  $\|x\|$  is used to denote an arbitrary  $p$ -norm; it will be made clear when a specific norm is considered. For a vector  $x \in \mathbb{R}^n$ , define  $|x| := [|x_1| \dots |x_n|]^T$ . Also for a vector  $x \in \mathbb{R}^n$ , by  $\max |x|$  we denote

$\max\{|x_1|, \dots, |x_n|\}$ . For a scalar  $x \in \mathbb{R}$ , denote by  $\lceil x \rceil$  the smallest integer number larger than  $x$ .

A function  $\alpha : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  is said to belong to class  $\mathcal{K}$ , i.e.,  $\alpha \in \mathcal{K}$ , if it is continuous, strictly increasing and  $\alpha(0) = 0$ . Furthermore,  $\alpha \in \mathcal{K}_\infty$  if  $\alpha \in \mathcal{K}$  and  $\lim_{s \rightarrow \infty} \alpha(s) = \infty$ .

### 2.2 Suboptimal MPC problem formulation

Let us consider the discrete-time system described by

$$x_{k+1} = f(x_k, u_k), \quad (1)$$

where  $x_k \in \mathbb{R}^n$  is the state and  $u_k \in \mathbb{R}^m$  is the control vector at discrete-time  $k \in \mathbb{Z}_+$ . We assume that the map  $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$  satisfies  $f(0, 0) = 0$ , which is, the origin is an equilibrium point for system (1).

The goal of MPC is to regulate the state to the origin while satisfying control and state constraints, i.e.,  $u_k \in \mathbb{U} \subset \mathbb{R}^m$  and  $x_k \in \mathbb{X} \subset \mathbb{R}^n$  for all  $k \in \mathbb{Z}_+$ , where  $\mathbb{U}$  and  $\mathbb{X}$  are proper sets. MPC relies on a receding-horizon control law in order to determine, for each  $k$ , a finite-sequence of control inputs

$$U(k) = \{u_{k|k}, u_{k+1|k}, \dots, u_{k+N-1|k}\},$$

where  $N$  is the control and prediction horizon, which are considered equal in this paper, for simplicity of exposition. If the initial state is  $x_{k|k} = x_k$  and the control sequence is  $U(k)$ , the solution of system (1) in closed-loop with  $U(k)$  at time  $k+i$  is denoted by  $\phi(x_{k|k}, U(k), i)$ . The current control action  $u_k$ , is selected as the first control action in  $U(k)$ , i.e.,  $u_k = u_{k|k}$ .

To achieve this, optimal MPC minimizes, at each discrete-time  $k$ , a cost function of the type

$$J(x_{k|k}, U(k)) = V_f(x_{k+N|k}) + \sum_{i=0}^{N-1} L(x_{k+i|k}, u_{k+i|k}), \quad (2)$$

where  $J : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}_+$  is the total cost function,  $V_f : \mathbb{R}^n \rightarrow \mathbb{R}_+$  is a terminal cost and  $L : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}_+$  is a stage cost. The minimization is performed with respect to  $U(k)$  and it is subject to

$$x_{k+j|k} = f(x_{k+j-1|k}, u_{k+j-1|k}), \quad \forall j \in \mathbb{Z}_{[1, N]}, \quad (3)$$

and to the state and input constraints.

When the dynamics  $f$  is a nonlinear, possibly non-convex function, the optimization of the cost (2) cannot be guaranteed to converge to a global optimum, in general. Alternatively, suboptimal MPC was proposed as a viable alternative, see, e.g., (Scokaert et al., 1999), to deal with this inherent shortcoming of nonlinear global optimization.

Assume that the following constraints are required to hold at each iteration of the MPC problem:

$$x_{k+i|k} \in \mathbb{X}, u_{k+i|k} \in \mathbb{U}, \quad \forall i \in \mathbb{Z}_{[0, N-1]}, \quad (4)$$

and

$$x_{k+N|k} \in \mathbb{X}_T, \quad (5)$$

where  $\mathbb{X}_T \subseteq \mathbb{X}$  is a proper set which represents a terminal constraint. Moreover, define by  $\mathcal{U}(x_{k|k})$  the set of control sequences  $U(k)$  which, applied on  $x_{k|k}$ , satisfy (3), (4) and (5).

Suboptimal MPC relies on an initial feasible solution, a warm start sequence  $U_{warm}(k) \in \mathcal{U}(x_{k|k})$  at each step  $k$ , which is improved iteratively. The suboptimal MPC problem considered in this paper is formulated as follows:

*Problem 2.1.* For each  $k \in \mathbb{Z}_{\geq 1}$ , given  $U_{warm}(k)$  find a sequence  $U(k) \in \mathcal{U}(x_{k|k})$  such that

$$J(x_{k|k}, U_{warm}(k)) > J(x_{k|k}, U(k)), \quad (6)$$

and the constraints (3), (4) and (5) are satisfied.

Consider now a locally stabilizing control law  $k_f : \mathbb{X}_T \rightarrow \mathbb{U}$ . Assume that  $\mathbb{X}_T$  is a sublevel set of  $V_f$ . For stability of the MPC closed-loop system it is also required that:

- $V_f(f(x, k_f(x))) + L(x, k_f(x)) \leq V_f(x)$  for all  $x \in \mathbb{X}_T$ ;
- there exist  $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$  such that  $\alpha_1(|x|) \leq V_f(x) \leq \alpha_2(|x|)$  for all  $x \in \mathbb{X}_T$ ;
- there exist  $\alpha_3 \in \mathcal{K}_\infty$  such that  $L(x, u) \geq \alpha_3(|x|)$  for all  $(x, u) \in \mathbb{X} \times \mathbb{U}$ ;

*Remark 2.2.* The first property above listed implies that  $\mathbb{X}_T$  is positively invariant for the system  $x_{k+1} = f(x_k, k_f(x_k))$ . The second and third properties can be satisfied if, for example  $V_f$  and  $L$  are positive definite quadratic functions.

These properties imply that the cost function  $J(\cdot, \cdot)$  is a Lyapunov function, see (Mayne and Rawlings, 2009, Lemma 2.14). As such, if  $\mathbb{F}$  is the set of states in  $\mathbb{X}$  for which there exists a control sequence  $U(k)$  which satisfies the constraints (3), (4) and (5), then the solution to Problem 2.1 provides an asymptotically stabilizing controller with a region of attraction  $\mathbb{F}$ .

In (Mayne and Rawlings, 2009), an algorithm was proposed for suboptimal MPC with stability guarantees. Given the current state  $x_{k|k}$  and the previous control sequence  $U(k-1)$  as an input, the steps of the algorithm therein can be summarized as follows:

- If  $x_{k|k} \notin \mathbb{X}_T$ , use the warm start sequence:

$$U_{warm}(k) = \{u_{k|k-1}, u_{k+1|k-1}, \dots, u_{k+N-2|k-1}, k_f(x_{k+N-1|k-1})\}. \quad (7)$$

Solve iteratively Problem 2.1 via optimization to improve  $U_{warm}(k)$  with a  $U(k) \in \mathcal{U}(x_{k|k})$ . Apply control  $u_k = u_{k|k}$ .

- If  $x_{k|k} \in \mathbb{X}_T$ , set  $u_{k|k} = k_f(x_{k|k})$ , or, similarly to the previous case, use the warm start and solve an optimization algorithm iteratively to find an improved control sequence  $U(k) \in \mathcal{U}(x_{k|k})$ .

Optimization solvers, in both optimal and suboptimal MPC, present difficulties in terms of parallelization and a priori known execution time independently of the current state  $x_k$ . To circumvent these drawbacks and enable a computationally efficient and parallelizable nonlinear MPC algorithm, we propose a sampling based approach to solving Problem 2.1, as illustrated in Section 3. To this end, we first review existing, similar approaches within nonlinear MPC.

### 2.3 Existing approaches based on sampling

This section provides a brief in depth review of two main existing approaches for NMPC based on sampling, namely approximate DP and sampling-based MPC, which were also mentioned in the Introduction.

An approximate version of DP, as a tool for solving optimization problems, was proposed in (Bertsekas, 2005a, Section 6.6.1). DP has been successfully applied for determining explicit solutions for linear MPC controllers, see, e.g., (Muñoz de la Peña et al., 2004). In nonlinear MPC, the state and input

states are typically discretized to apply DP algorithms. The main idea is to discretize the state space with a finite grid and to express each state outside of the grid as an interpolation of nearby grid elements. The same interpolation law is applied to compute the cost of the current nongrid state as a function of the costs of the nearby grid states. As such, by discretizing both the state space for each time in the control horizon and the input space, a transition diagram is obtained which approximates the dynamics of the system in the continuous space. On this discrete transition system, DP is applied to determine the path with the smallest cost, which, for a given initial state  $x_{k|k}$ , provides the control sequence  $U(k)$ .

Solving MPC with DP via discretization suffers from the “curse of dimensionality”, due to sampling of both state and input spaces and the requirement for constructing the complete transition diagram, by evaluating the subsequent state and cost for each sampled state and all the samples in the input space.

An alternative to approximate DP, namely sampling based MPC, was developed within the area of Robotics, where typically optimization problems arising in control are non-convex, due to either kinematic constraints or constraints posed by obstacle avoidance. Sampling-based motion planning such as rapidly-exploring random trees (RRTs) (LaValle, 1998) or randomized A\* algorithms (Likhachev and Stentz, 2008), have been extensively used to construct trees which connect an initial state to a final state based on sampling states in the search space and searching for feasible inputs to connect these states. To approach issues related to the, possibly unfeasible, search for an input after sampling only in the state space, a method (SBMPC) was proposed in (Dunlap et al., 2010), which samples the input space at each sampling period and creates trees that contain feasible state trajectories. The optimal path to a goal in the state-space is then searched for within the tree using goal-directed search algorithms, such as LPA\*. Such algorithms rely on computing a heuristic measure of the distance from the current sample to the goal. Selecting the heuristic is however, not an obvious task for general nonlinear systems.

Therefore, a desirable feature of a sampling-based suboptimal NMPC algorithm is a non-exponential growth in the tree generated through sampling of the state or input space at each step in the horizon. Furthermore, it is also desirable to reduce the dependency of the algorithm on the non-obvious selection of a heuristic, which significantly impacts the performance of the sampling-based strategy. To circumvent these issues, an alternative suboptimal strategy for sampling the input space is proposed in the next section, based on sequentially updating a warm start feasible sequence of predicted inputs.

## 3. MAIN RESULT

In this section we present the proposed sampling-based algorithm for solving suboptimal NMPC problems and we provide a detailed complexity analysis.

### 3.1 Prototype algorithm

Inspired by the suboptimal MPC and DP principles, we propose a sampling-based solution to Problem 2.1, which, by the mechanism involved in the iterative improvement of the initial feasible control sequence  $U_{warm}(k)$ , has a low increase with the control horizon of the computational complexity.

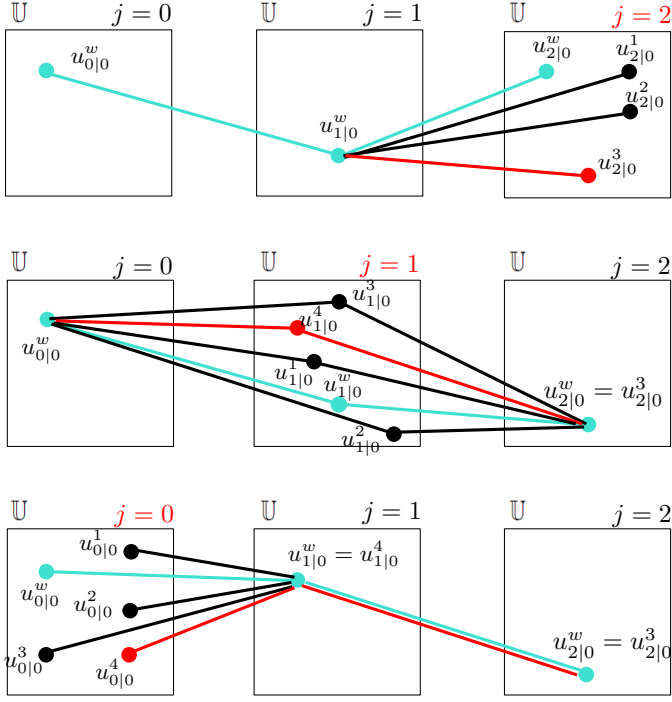


Fig. 1. Sampling principle.

---

**Algorithm 1** Sampling-based suboptimal NMPC algorithm.

---

**Input:**  $N, \{n_j\}_{j \in \mathbb{Z}_{[0, N-1]}}, x_{k|k}, \mathbb{X}, \mathbb{U}, \mathbb{X}_T, J(\cdot, \cdot)$

$$U_{warm}(k) = \{u_{k|k}^w, \dots, u_{k+N-1|k}^w\}$$

**Output:**  $U(k), u_k$

```

1:  $J_{sub} \leftarrow J(x_{k|k}, U_{warm}(k));$ 
2: for all  $j = N - 1 : -1 : 0$  do
3:   Select  $n_j$  samples  $u_{j+k|k}^q \in \mathbb{U}, q \in \mathbb{Z}_{[1, \dots, n_j]}$ ;
4:   for all  $q = 1 : 1 : n_j$  do
5:     if  $j \geq 1$  then
6:        $U(k)^{jq} = \{u_{k|k}^w, \dots, u_{k+j-1|k}^w, u_{k+j|k}^q,$ 
          $u_{k+j+1|k}^w, \dots, u_{k+N-1|k}^w\};$ 
7:     else
8:        $U(k)^{jq} = \{u_{k|k}^q, u_{k+1|k}^w, \dots, u_{k+N-1|k}^w\};$ 
9:     if  $\phi(x_{k|k}, U(k)^{jq}, i) \in \mathbb{X}, \forall i \in \mathbb{Z}_{[j+1, N-1]}$  and
        $\phi(x_{k|k}, U(k)^{jq}, N) \in \mathbb{X}_T$  then
10:       $J_{new} \leftarrow J(x_{k|k}, U(k)^{jq});$ 
11:      if  $J_{new} < J_{sub}$  then
12:         $J_{sub} \leftarrow J_{new};$ 
13:         $U_{warm}(k) \leftarrow U(k)^{jq};$ 
14:  $U(k) = U_{warm}(k), u_k = u_{k|k}^w;$ 
```

---

The principle behind the proposed sampling-based approach is illustrated in Fig. 1 and formalized in Algorithm 1. In Fig. 1, the iterative improvement of an initial cost provided by an initial feasible control sequence  $U_{warm}(k)$  is illustrated for the case when  $N = 3$ . The algorithm keeps always  $U_{warm}(k)$  as a reference sequence, and it covers the horizon in a backward fashion, in  $N$  iterations. Starting with  $j = N - 1$ , at each iteration step,  $n_j$  samples  $\{u_{k+j|k}^q\}_{q \in \mathbb{Z}_{[1, n_j]}}$  are drawn from the input constraint set  $\mathbb{U}$ . For each sample, the reference sequence  $U_{warm}(k)$  is modified in the  $j^{th}$  location, and a new sequence,  $U(k)^{jq}$  is obtained. If the following properties hold:

- $U(k)^{jq}$  is a feasible sequence, i.e., the constraints (3), (4) and (5) hold,
- the new cost,  $J_{new} = J(x_{k|k}, U(k)^{jq})$  decreases with respect to  $J(x_{k|k}, U_{warm}(k))$ ,

then  $U_{warm}(k)$  is replaced by  $U(k)^{jq}$  and the algorithm continues backwards with respect to the prediction time  $j$ , in a similar manner. With this approach, at any point in time, if the maximally allowed computational time is exceeded, a feasible, improved control sequence exists and it can be utilized as a suboptimal MPC solution.

By choosing to cover the control horizon backwards, we can reuse at each step  $j$ , the states  $\phi(x_{k|k}, U_{warm}(k), i)$  for all  $i \in \mathbb{Z}_{[1, j]}$ , which, by the feasibility of  $U_{warm}(k)$ , already satisfy the state and input constraints. This holds not only for the original  $U_{warm}(k)$ , but for any subsequent improvement of  $U_{warm}(k)$ . As such, also the stage costs up to the  $j^{th}$  state can be recovered from previous computations. The reusability of previously computed costs and states by navigating the control horizon in a backward manner resembles the working principle of DP. This suggests intuitively that the proposed cost improvement method could deliver good performance, which is supported by results obtained in non-trivial case studies, see Section 4. A formal analysis of convergence towards the DP solution, as the discrete-time  $k$  increases, makes the object of future research.

When  $k = 0$ , we can select an initial sequence  $U_{warm}(0)$  as the solution of the optimal MPC problem. In this case, we can proceed with  $k = 1$ . Alternatively, we can select a feasible sequence  $U_{warm}(0)$ , an “oracle”, by randomly selecting sequences of inputs in  $\mathbb{U}$  until a feasible  $U_{warm}(0)$  is found. In this case, if it is feasible to afford the computational time, we can proceed with Algorithm 1 in an attempt of obtaining an improved sequence.

For  $k \in \mathbb{Z}_{\geq 1}$ , to choose the input  $U_{warm}(k)$  for Algorithm 1, one can use the receding horizon principle of MPC. As such, the input sequence  $U_{warm}(k) = \{u_{k|k-1}, \dots, u_{k+N-2|k-1}, u\}$  is a warm start at time  $k$ . If  $\phi(x_{k-1|k-1}, U(k-1), N) \in \mathbb{X}_T$  and  $\mathbb{X}_T$  is positively invariant for the system  $x_{k+1} = f(x_k, k_f(x_k))$ , one can select  $u = k_f(x_{k+N-1|k-1})$ . In this case,  $U_{warm}(k)$  is a feasible solution, and therefore a candidate warm start for every  $k \in \mathbb{Z}_{\geq 1}$ . If there exists no terminal set  $\mathbb{X}_T$  and no  $k_f(\cdot)$ , then one can select  $u \in \mathbb{U}$  such that  $U_{warm}(k)$  remains feasible, i.e.,  $\phi(x_{k-1|k-1}, U_{warm}(k), N) \in \mathbb{X}$ . In these circumstances, however, stability of the closed loop is not guaranteed. Such an example is illustrated in Section 4.3.

**Remark 3.1.** Common sampling schemes are employed for sampling of the input space  $\mathbb{U}$  at each iteration, among which we consider deterministic uniform sampling, which places each sample at equal distance from each other, to cover uniformly the space  $\mathbb{U}$ . Alternatively, “true” random samples can be selected, which are simpler to draw in higher dimensional spaces. Quasi-random low-discrepancy sequences, see (Chakrabarty et al., 2016), may be used as well, considering the fact that they appear to be random for multiple purposes, such as Monte Carlo simulations. Such sampling methods, e.g., Sobol or Halton sequences, have been shown, see, e.g., (de Dios Ortúzar et al., 1994), to better cover the space than random sequences.

**Remark 3.2.** Assume that  $k_f$  is a locally stabilizing control law on  $\mathbb{X}_T$ , a sublevel set of  $V_f$ , which is positively invariant for the system  $x_{k+1} = f(x_k, k_f(x_k))$  and  $V_f$  and  $L$  are, e.g., positive

definite quadratic functions. Considering that the sequence  $U(k)$  provided by Algorithm 1 is a suboptimal solution solving Problem 2.1, then, by the arguments in Section 2.2, Problem 2.1 is recursively feasible and it ensures stability of system (1) in closed loop with  $u_k = u_{k|k}^w$ .

*Remark 3.3.* The working mechanism of Algorithm 1 resembles the working principle of the rollout algorithm described in the survey paper (Bertsekas, 2005b). Therein, a rollout algorithm improves iteratively a base policy (here, the feasible control sequence  $U_{warm}(k)$ ) to provide a suboptimal solution to an optimal control problem via approximative DP and suboptimal control. The working principle of the rollout algorithm with sampling based MPC, i.e., choosing at time  $k$  the iterated solution of the previous time instance,  $k - 1$ , as a warm start, and a sampling strategy, is not provided therein.

*Remark 3.4.* Due to the sampling procedure having a discontinuous behaviour, the inputs might be varying more than it is safe for some applications. This problem might be alleviated by either smoothening the inputs via interpolation of the obtained input sequence with the initial sequence, or by penalizing  $\Delta u_k = u_k - u_{k-1}$  via constraints or via the cost function, such that the input variability is limited to acceptable bounds.

### 3.2 Complexity analysis

In order to analyze the complexity of Algorithm 1, the following assumptions are undertaken, for a given state  $x$ , input  $u$  and input sequences  $U, U_1, U_2$ :

- 1) The cost of evaluating  $f(x, u)$  and performing a feasibility test  $f(x, u) \in \mathbb{X}$  is  $c_1$ ;
- 2) The cost of evaluating  $J(x, U)$  is  $c_2$ ;
- 3) The cost of comparing  $J(x, U_1) < J(x, U_2)$  and changing  $J_{sub}$  and  $U_{warm}(k)$  if necessary, i.e., steps 11-13 in Algorithm 1, is negligible.
- 4) The current cost  $J_{sub}$  is instantaneously available for comparison with each of the  $n_j$  samples according to step 11 in Algorithm 1 and each of the new sequences  $U(k)^{jq}$  may modify  $J_{sub}$  if the new cost  $J_{new}$  is smaller than  $J_{sub}$ .

The complexity of Algorithm 1 for a given  $x_{k|k}$  is the following:

$$C = c_1 \left( \sum_{j=0}^{N-1} (N-j)n_j \right) + c_2 \left( \sum_{j=0}^{N-1} n_j \right). \quad (8)$$

If we assume  $n_j \leq \bar{n}$  for all  $j \in \mathbb{Z}_{[0, N-1]}$ , then the complexity (8) can be upper bounded by

$$C = \bar{n}c_1 \frac{N(N+1)}{2} + c_2 N\bar{n}. \quad (9)$$

A possible reduction of the bound (9) might be attained, considering the fact that all the states subsequent to a non-feasible state are no longer evaluated and checked for feasibility. This means that, in step 9 of Algorithm 1, if  $\phi(x_{k|k}, U(k)^{jq}, i) \notin \mathbb{X}$  for a specific  $i \in \mathbb{Z}_{[j+1, N-1]}$ , then  $\phi(x_{k|k}, U(k)^{jq}, r)$  for all  $r \in \mathbb{Z}_{[i+1, N]}$ , are no longer evaluated, in which case steps 2) and 3) are skipped all together.

Consider now that many threads are available, from multiple processors. Notice also that at each time in the horizon, all  $n_j$  computations can be performed separately. In these conditions, assuming we have  $\bar{n}$  threads available, then the complexity of Algorithm 1 is upper bounded by

$$C = c_1 \frac{N(N+1)}{2} + c_2 N. \quad (10)$$

The complexity bound given in (10), though quadratic in the prediction horizon, yields a reasonable complexity, considering that, in NMPC, a horizon  $N = 10$  is considered a reasonably large horizon.

In general, if we have  $p \in \mathbb{Z}_{[2, \bar{n})}$  processors, then the complexity of Algorithm 1 is

$$C = \lceil \bar{n}/p \rceil \left( c_1 \frac{N(N+1)}{2} + c_2 N \right), \quad (11)$$

where the term  $\lceil \bar{n}/p \rceil$  appears due to the fact that a thread can not engage in computations related to a subsequent time horizon until all the threads have finalized the computations related to the current time horizon  $j$ .

## 4. ILLUSTRATIVE EXAMPLES

The sampling-based suboptimal NMPC strategy proposed in Section 3 is illustrated on three nonlinear systems, to highlight various features of this method. All tests have been performed on a system with the following specifications: Intel Core i7-3770 CPU 3.4GHz, 16GB RAM, 64-bit OS.

### 4.1 Cart-spring system

The method developed in this paper will first be applied to a system incorporating an exponential nonlinearity, i.e., the model of a cart with mass  $M$ , which is moving on a plane, see (Raimondo et al., 2009). This cart is attached to a wall via a spring with elastic constant  $k$  varying with the first state  $k = k_0 e^{-x_1}$ , where  $x_1$  stands for the displacement of the carriage from the equilibrium position. A damper acts as a resistor in the system, with damping  $h_d$ . The discretized nonlinear model of the cart and spring system is the following:

$$x(k+1) = \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = f_1(x(k)) + F_2 u(k), \quad (12)$$

where

$$f_1(x(k)) = \begin{bmatrix} x_1(k) + T_s x_2(k) \\ x_2(k) - T_s \frac{\rho_0}{M} e^{-x_1} x_1(k) - T_s \frac{h_d}{M} x_2(k) \end{bmatrix}, \quad (13)$$

$$F_2 = \begin{bmatrix} 0 & T_s \\ 0 & T_s \end{bmatrix}^T,$$

where  $x_2$  is the velocity of the cart and  $u$  is an external force which acts as an input to the system. The parameter values are  $T_s = 0.4s$ ,  $\rho_0 = 0.33$ ,  $M = 1$ ,  $h_d = 1.1$ .

The MPC controller has to steer the cart to the origin from a non-zero initial state, while satisfying the input and state constraints, which are

$$|u| \leq 4.5, |x_1| \leq 2.65, \quad (14)$$

and reducing the cost (2), where the stage cost and terminal cost are quadratic functions, i.e.,  $L(x, u) = x^T Q x + u^T R u$  and  $V_f(x) = x^T P x$ . Choose the following parameters for the MPC problem:

$$M = 4, P = \begin{bmatrix} 7.0814 & 3.3708 \\ 3.3708 & 4.2998 \end{bmatrix}, Q = \text{diag}(1, 1), R = 1.$$

In (Raimondo et al., 2009) it is shown that the control law

$$u = k_f(x) = -[0.8783 \ 1.1204] f_1(x),$$

is locally stabilizing in the set

$$\mathbb{X}_T = \{x | V_f(x) \leq 4.7\},$$

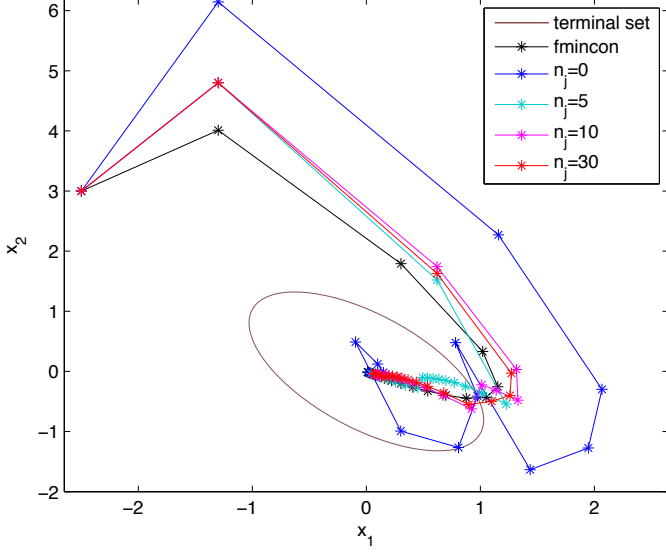


Fig. 2. State trajectory.

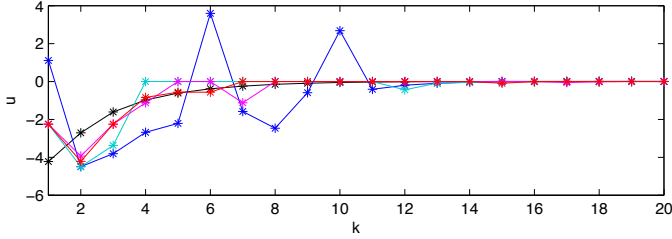


Fig. 3. Inputs applied to the system.

which is a terminal set where the conditions for stability of MPC mentioned in Section 2.2 are satisfied.

Algorithm 1 is applied for the MPC control of system (12). We compare the results of this method with the results provided by `fmincon` in Matlab, even though, as it will be seen later, the optimization tool does not always provide the optimum, due to local minima. The scalability of the algorithm is tested by varying both the number of samples  $\bar{n}$  and the control horizon  $N$ . The tests illustrated in this paper are performed on a feasible initial state  $x_{0|0} = [-2.5, 3]$ . The choice of the initial condition does not influence greatly the results, which are similar for other feasible initial states.

For the first experiment, fix  $N = 10$ . An initial  $U_{warm}(0)$  is provided by a random “oracle”. Consider  $n_j = \bar{n}$ , for all  $j \in \mathbb{Z}_{[0, N-1]}$ , taking various values in the set  $\{0, 5, 10, 30\}$ . For  $\bar{n} = 0$ ,  $U_{warm}(0)$  is propagated through iterations without any intervention or change from the sampling mechanism. This serves as a reference, to notice the improvements brought in by Algorithm 1. The results are illustrated in Fig. 2–5, where the legend from Fig. 2 holds until Fig. 5. Iterations are considered from  $k = 1$  until  $k = 20$ . Though the different sampling options proposed in Remark 3.1 provide in this example similar outcomes, Halton points have been used here for illustration. This choice is motivated by the practical feature that, adding extra points only when they are required does not have impact on the coverage of the set  $\mathbb{U}$ .

In Fig. 2 and Fig. 3 the state trajectory and the inputs  $u(k)$  applied to the system are illustrated. The constraint specifications (14) are satisfied for all cases, at all times. Notice the

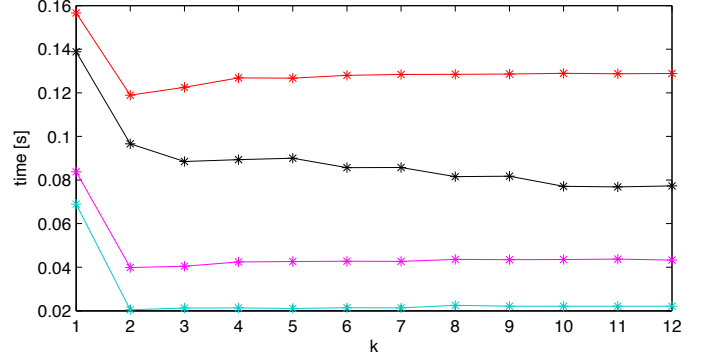


Fig. 4. Computation time.

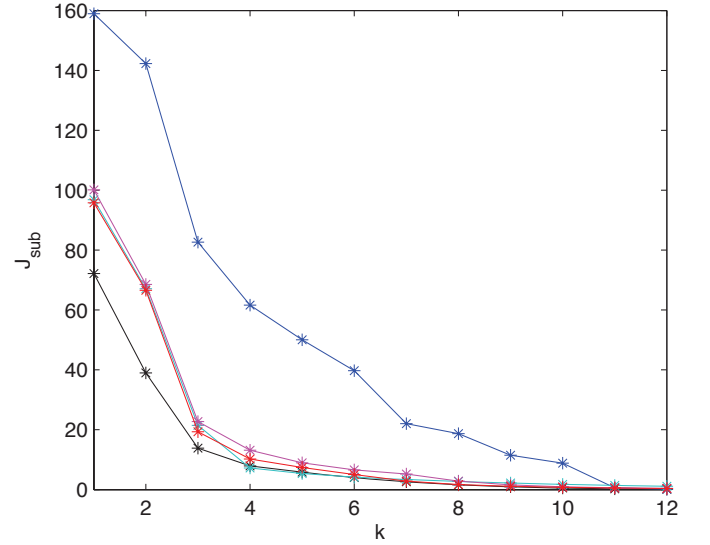


Fig. 5. The cost  $J_{sub}$ .

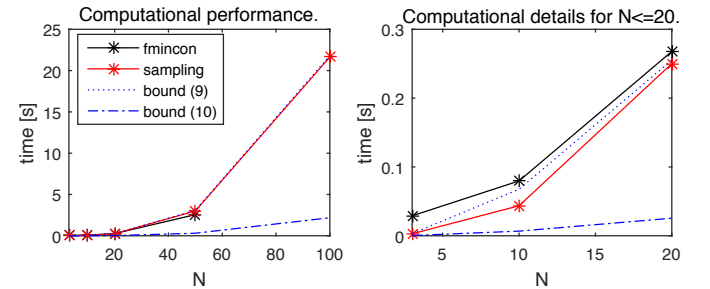


Fig. 6. Computational performance versus proposed bounds.

immediate smoothing of the trajectories and input sequence even for a small  $\bar{n}$ . In Fig. 4, the computational time, without parallelization, is illustrated. At  $k = 1$ , the computational cost of finding an oracle is included. Fig. 5 illustrates the values of  $J_{sub}$  for each iteration. Notice, overall, that even for a small number of samples, the performance of the closed loop system is significantly improved and the computational time is promising, even for a non-parallel implementation. Also, as the iteration  $k$  advances, the initially modest performance increases significantly, due to the continuous improvement of  $U_{warm}$  and the receding horizon principle. Interestingly, even for  $\bar{n} = 5$ , at iteration  $k = 4$ , the cost  $J_{sub}(4)$  is smaller than the cost computed via `fmincon`, which, due to local minima, provided a feasible but not optimal solution.



For the second experiment we aim to test the computational complexity of Algorithm 1 in terms of the control horizon. The implementation used here does not use parallelization. Fix  $\bar{n} = 10$  and  $N$  takes values in the set  $\{3, 10, 20, 50, 100\}$ . The results are illustrated in Fig. 6. With red, the computational complexity for Algorithm 1 is depicted, and it is always smaller than the bound (9), which is followed closely. It is expected that on dedicated devices, the complexity of Algorithm 1 is smaller, due to the processors not running in parallel threads related to other system applications. With parallelization, further reduction in complexity is expected, as described in Section 3.2 and illustrated in Fig. 6. Notice that, for smaller horizons, the complexity of Algorithm 1 is smaller than the complexity of `fmincon`, even without parallelization. For  $N = 100$ , `fmincon` provides solutions which are not feasible, while the proposed Algorithm 1 still terminates in 21.6 seconds.

#### 4.2 Buck-Boost power converter

Next, the bilinear model of a Buck-Boost power converter is considered, as in (Spinu et al., 2011):

$$x(k+1) = Ax(k) + Bu(k) + \begin{bmatrix} x(k)^T C_1 \\ x(k)^T C_2 \end{bmatrix} u(k), \quad (15)$$

where  $x := [v_C \ i_L]^T \in \mathbb{X} \subset \mathbb{R}^2$  is the state vector consisting of the voltage across the output capacitor and the current through the filter inductor. The input  $u := [d_1 \ d_2]^T \in \mathbb{U} \subset \mathbb{R}^2$  stands for the duty-cycle ratio of the control signal applied to the switching node. The parameters are

$$A = \left( I_2 + T_s \begin{bmatrix} -\frac{1}{R_H C} & 0 \\ 0 & -\frac{R_L}{L} \end{bmatrix} \right), B = \begin{bmatrix} 0 & 0 \\ \frac{v_s}{L} & 0 \end{bmatrix} T_s, \\ C_1 = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{C} \end{bmatrix} T_s, C_2 = \begin{bmatrix} 0 & -\frac{1}{L} \\ 0 & 0 \end{bmatrix} T_s,$$

with the values  $R_L = 0.2\Omega$ ,  $C = 22\mu F$ ,  $L = 220\mu H$ ,  $T_s = 10\mu s$ .

The aim of the control loop is to stabilize the system to the equilibrium point  $x_e := [20 \ 0.5]^T$ ,  $u_e := [0.81 \ 0.4]^T$ , under the constraints  $i_L \in \mathbb{R}_{[0,3]}$ ,  $v_C \in \mathbb{R}_{[-0.1,22.5]}$ ,  $u \in \mathbb{R}_{[0,1]}^2$ . The terminal controller

$$u = u_e + K(x - x_e), K = \begin{bmatrix} -0.0014 & -0.3246 \\ 0.0001 & -0.0055 \end{bmatrix},$$

stabilizing the system in the terminal set  $\mathbb{X}_T$  given in (Spinu et al., 2011, set  $\mathbb{P}$  in Section IV.C), and the quadratic cost with the matrices

$$Q = \text{diag}(1, 2), R = \text{diag}(1, 1), P = \begin{bmatrix} 46.6617 & 42.8039 \\ 42.8039 & 69.4392 \end{bmatrix},$$

satisfies all the conditions for stability formulated in Section 2.2.

Similarly to the previous example, we apply Algorithm 1 for the MPC control of system (15). Fix the initial state  $x_{0|0} = [1 \ 2]^T + x_e$ , which is outside of the terminal set  $\mathbb{X}_T$ , and  $N = 10$ .  $U_{warm}(0)$  is given by an oracle and we use random sampling of  $\mathbb{U}$ . See in Fig. 7 the effect of varying  $\bar{n}$  on the cost function  $J_{sub}$  and the time necessary, per iteration, to compute the corresponding control law. Notice the effect of the unknown termination time on the evaluation time for the optimization performed through `fmincon` and the relatively equal computational time of Algorithm 1 over the iterations  $k$ .

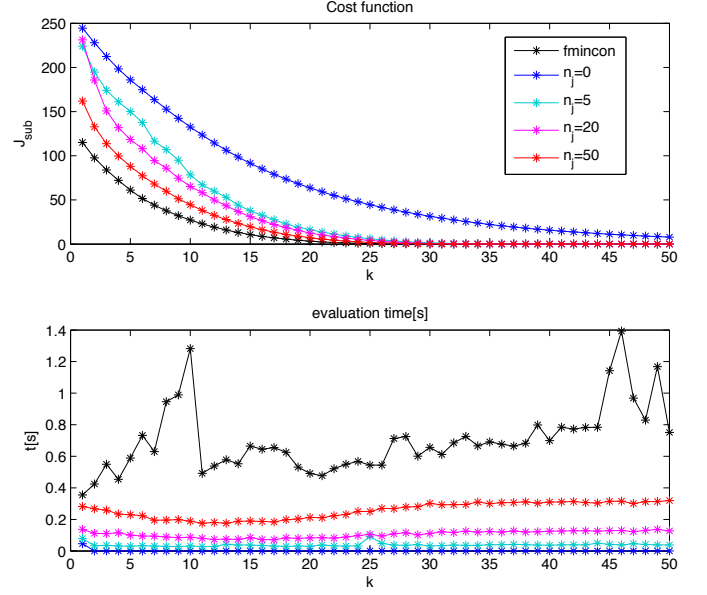


Fig. 7. Performance of Algorithm 1 on the power converter.

All the constraints were satisfied for all the presented situations, and it is expected that, through implementation on dedicated multi-thread systems, the computational time for the control law decreases to the extent of fitting the tight sampling period of the power converter.

#### 4.3 Wheeled mobile robot

The last example illustrates the methodology developed in this paper for an obstacle avoidance task by a nonholonomic system with trigonometric nonlinearities, due to kinematics, i.e., a model of the wheeled mobile robot (WMR), as described in (Kuhne et al., 2005):

$$x(k+1) = \begin{bmatrix} x_1(k) + u_1(k) \cos x_3(k) T_s \\ x_2(k) + u_1(k) \sin x_3(k) T_s \\ x_3(k) + u_2(k) T_s \end{bmatrix}. \quad (16)$$

In (16), the state  $x \in \mathbb{R}^3$  describes the position and the orientation of the robot with respect to a global inertial frame  $\{O, X, Y\}$ , and the input  $u \in \mathbb{R}^2$  gives the linear and angular velocity, respectively. The parameter  $T_s = 0.1s$  is the discretization period of system (16).

The MPC strategy aims at driving the WMR from an initial state  $x_{0|0} = [0 \ 6 \ 0]^T$  to the origin of the inertial frame, i.e.,  $x_g = [0 \ 0 \ 0]^T$ , while satisfying the input constraints  $u_1 \in \mathbb{R}_{[-0.47, 0.47]}$ ,  $u_2 \in \mathbb{R}_{[-3.77, 3.77]}$ . A quadratic cost function of the form

$$J(x_{k|k}, U(k)) = x_{k+N|k}^T P x_{k+N|k} + \sum_{j=1}^{N-1} x_{k+j|k}^T Q(j) x_{k+j|k} + \sum_{j=0}^{N-1} u_{k+j|k}^T R u_{k+j|k}$$

is considered, with the parameters:  $Q(j) = 2^{j-1}Q$ ,  $P = 50Q(N)$ ,  $Q = \text{diag}(1, 1, 0.5)$ ,  $R = \text{diag}(0.1, 0.1)$ ,  $N = 5$ .

In this example we illustrate Algorithm 1 with the above parameters. Consider  $U_{warm}(0)$  generated by an “oracle”, and sampling of  $\mathbb{U}$  based on random sequences. The result for  $\bar{n} = 30$  is illustrated in Fig 8. `fmincon` could not be applied

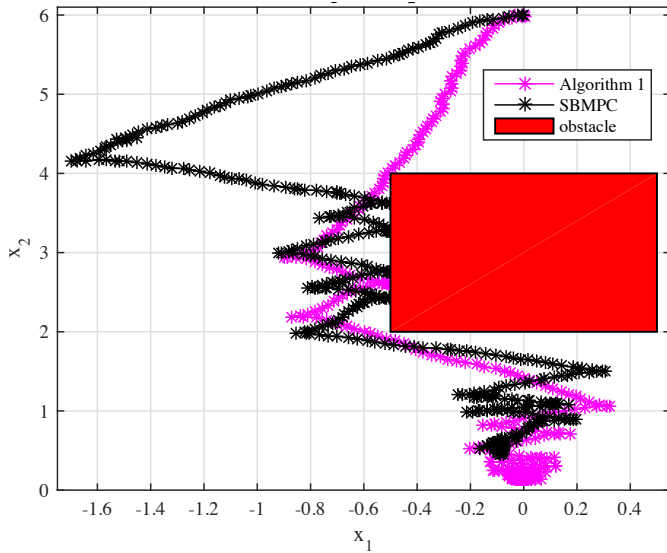


Fig. 8. Trajectories of WMR.

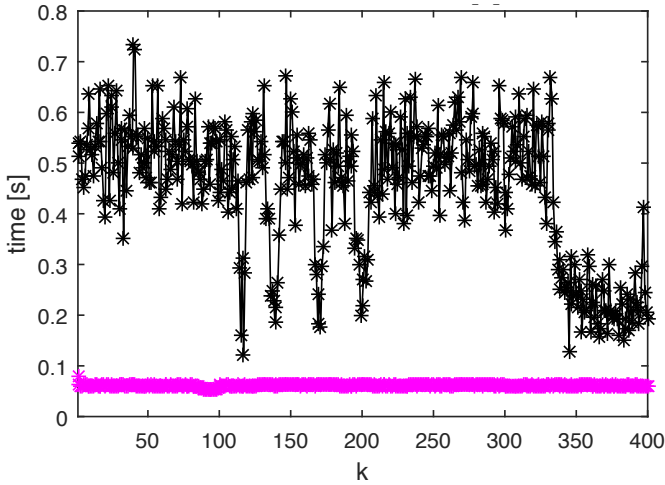


Fig. 9. Time performance of WMR.

due to feasibility issues related to non-convexity caused by the presence of the obstacle. Notice the avoidance of the obstacle of the WMR under the control law generated by Algorithm 1. Due to the sampling mechanism, however, the inputs are not smooth, which could be alleviated, e.g., by an interpolation mechanism. For comparison we illustrate also the result using SBMPC (Dunlap et al., 2010), with the same horizon  $N$  and  $\bar{n} = 30$ . Notice, in the state trajectory plot in Fig 8 the fact that, after 400 iterations, the WMR did not reach  $x_g$  yet, and the input values are not yet 0, which means that the sampling mechanism did not consider that the WMR is close to the goal. In this case, it is recommended to sample more densely around 0 in the set  $\mathbb{U}$ , rather than uniformly covering  $\mathbb{U}$  with samples.

Notice in Fig 9 that, due to the building of a tree and the search in a tree for the best path, SBMPC is more computationally demanding. The complexity of Algorithm 1 with the given  $\bar{n}$  fits the sampling period  $T_s = 0.1s$  of the WMR, which makes the method applicable for real-time control of this system.

## 5. CONCLUSION

In this paper, an algorithm based on sampling of the input space at each time in the horizon was proposed, which iteratively

improves in terms of cost an initially feasible control sequence. This suboptimal NMPC strategy provides a promising computational complexity even for large control horizons, with good perspectives for parallel implementation. Future work aims at investigating the convergence of the proposed algorithm to the optimal solution, the scalability in terms of system dimension and the effect of parallelization on computational time for real-life applications.

## REFERENCES

- Bertsekas, D.P. (2005a). *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA.
- Bertsekas, D.P. (2005b). Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4), 310–334.
- Chakrabarty, A., Dinh, V., Corless, M., Rundell, A., Zak, S., and Buzzard, G. (2016). Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences. *IEEE Transactions on Automatic Control*, preprint published on-line, DOI: 10.1109/TAC.2016.2539222, PP(99).
- de Dios Ortúzar, J., Willumsen, L.G., et al. (1994). *Modelling transport*. Wiley New Jersey.
- Dunlap, D.D., Caldwell, C.V., and Collins, E.G. (2010). Non-linear model predictive control using sampling and goal-directed optimization. In *2010 IEEE International Conference on Control Applications*, 1349–1356. IEEE.
- Grüne, L. and Pannek, J. (2011). *Nonlinear model predictive control. Theory and algorithms*. London: Springer-Verlag.
- Johansen, T.A. (2004). Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, 40(2), 293–300.
- Kuhne, F., Lages, W.F., and Da Silva, J.G. (2005). Point stabilization of mobile robots with nonlinear model predictive control. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 3, 1163–1168. IEEE.
- LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. *TR 98-11, Computer Science Dept., Iowa State University*.
- Ławryńczuk, M. (2009). Computationally efficient nonlinear predictive control based on state-space neural models. In *International Conference on Parallel Processing and Applied Mathematics*, 350–359. Springer.
- Lazar, M. and Heemels, W. (2009). Predictive control of hybrid systems: Input-to-state stability results for sub-optimal solutions. *Automatica*, 45(1), 180–185.
- Lee, J.M. and Lee, J.H. (2004). Approximate dynamic programming strategies and their applicability for process control: A review and future directions. *International Journal of Control Automation and Systems*, 2, 263–278.
- Likhachev, M. and Stentz, A. (2008). R\* search. *Lab Papers (GRASP)*, 23.
- Magni, L., Raimondo, D.M., and Allgöwer, F. (2009). *Non-linear model predictive control: Towards new challenging applications*, volume 384. Springer, Lecture notes in Control and Information Sciences.
- Mayne, D. and Rawlings, J. (2009). *Model predictive control: theory and design*. Madison, WI: Nob Hill Publishing, LCC.
- Muñoz de la Peña, D., Alamo, T., Bemporad, A., and Camacho, E.F. (2004). A dynamic programming approach for determining the explicit solution of linear MPC controllers. In *IEEE Conference on Decision and Control*.



- Pannocchia, G., Rawlings, J.B., and Wright, S.J. (2011). Conditions under which suboptimal nonlinear MPC is inherently robust. *Systems & Control Letters*, 60(9), 747–755.
- Piovesan, J.L. and Tanner, H.G. (2009). Randomized model predictive control for robot navigation. In *IEEE International Conference on Robotics and Automation, 2009*, 94–99. Kobe, Japan.
- Raimondo, D.M., Limon, D., Lazar, M., Magni, L., and Camacho, E.F. (2009). Min-max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, 15(1), 5–21.
- Scokaert, P.O., Mayne, D.Q., and Rawlings, J.B. (1999). Sub-optimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3), 648–654.
- Spinu, V., Lazar, M., and van den Bosch, P. (2011). An explicit state-feedback solution to constrained stabilization of dc-dc power converters. In *2011 IEEE International Conference on Control Applications (CCA)*, 1112–1118. IEEE.